

Synthesis of Topological Quantum Circuits

Alexandru Paler¹

Simon Devitt²

Kae Nemoto²

Ilia Polian¹

¹Faculty of Informatics and Mathematics
University of Passau
Innstr. 43, D-94032 Passau, Germany
{alexandru.paler|ilia.polian}@uni-passau.de

²National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-ku
Tokyo, Japan
{devitt|nemoto}@nii.ac.jp

Abstract—Topological quantum computing has recently proven itself to be a very powerful model when considering large-scale, fully error corrected quantum architectures. In addition to its robust nature under hardware errors, it is a software driven method of error corrected computation, with the hardware responsible for only creating a generic quantum resource (the topological lattice). Computation in this scheme is achieved by the geometric manipulation of holes (defects) within the lattice. Interactions between logical qubits (quantum gate operations) are implemented by using particular arrangements of the defects, such as braids and junctions. We demonstrate that junction-based topological quantum gates allow highly regular and structured implementation of large CNOT (controlled-not) gate networks, which ultimately form the basis of the error corrected primitives that must be used for an error corrected algorithm. We present a number of heuristics to optimise the area of the resulting structures and therefore the number of the required hardware resources.

I. INTRODUCTION

Quantum information science has been one of the extraordinary success stories of theoretical and experimental physics in the last 20 years. Since the introduction of the field in the 1980s, there has been development of a complete theoretical framework for universal quantum computation and repeated experimental demonstration of quantum control on a small number of quantum bits (qubits) in multiple physical systems [1]. In addition to the experimental system development, there have been multiple quantum architectures proposed demonstrating how a large-scale multi-million qubit machine could be constructed [2]–[7].

Even with this level of experimental development, large scale quantum information processing requires hardware that is extraordinarily accurate; many orders of magnitude higher than any system has ever demonstrated. The inherent inaccuracies in quantum hardware induces errors during processing which must be corrected via dedicated and resource intensive Quantum Error Correcting (QEC) codes [8]. The topological model of QEC [9]–[11] has shown itself to be more promising to many other long-standing techniques and currently forms the basis of effectively all modern quantum-computing architectures [4]–[7]. The continuing success of experimental fabrication now requires us to seriously address the programming and operation of a large scale computer. This work is aimed at developing the required synthesis tools for large scale quantum information processing in the context of a fully error corrected system. We will present the classical framework for

converting the abstract circuits describing a quantum algorithm into the full set of error corrected operations applied by the actual hardware. This work represents arguably the first serious attempt to develop a classical framework for a fully error corrected computational model compatible with current architectural designs.

Topological Quantum Computing (TQC) is a model for universal computation that is constructed on the framework of active error correction. This method of computation very abstract when compared to classical computer science. One of the more bizarre aspects of this model is that the physical hardware *doesn't actually perform any real computation*. Instead, the hardware is only responsible for producing a very large three-dimensional lattice of qubits which are all linked together to form a single, massive quantum state. This quantum state forms the *workbench* of the computation and information is created, processed and read-out via the strategic manipulation of this massive quantum state produced from the hardware [12]. The volume of this lattice directly relates to the physical resources required for an error corrected algorithm. As large quantum algorithms commonly require thousands of encoded qubits and high levels of error correction are often needed, the volume of the lattice is an important criteria for the synthesis of large quantum circuits. Minimising this volume reduces the amount of physical resources required, and lowers the costs of computation [13].

In the topological model, quantum gates are realised via geometric shapes, known as defects, moving through the lattice (e.g. see Fig. 2). These realisations enable highly regular arrangements which are easily optimised using discrete algorithms. Unlike in [11] where error corrected gates are realised by moving defects around each other (known as braiding), we utilise a technique where gates are realised through *junctions* (where defects are jointed together in the lattice). These constructions of error corrected gates are minimal with regard to the occupied lattice volume, and enable an efficient placement within it.

This work presents compact representations for the necessary TQC quantum gates, and uses these for several heuristics that minimise the size of the required qubit lattice, known as the topological cluster. To the best of our knowledge, this is the first automatic quantum synthesis procedure with fully error corrected, TQC as the target technology.

II. BACKGROUND

A. Quantum Computing

Quantum computing can be, to a certain extent, described by building parallels to classical computing. A comprehensive review of quantum information and computation can be found in [14]. The concept of classical bit has its quantum counterpart, called a quantum bit (*qubit*). Like in classical computing, quantum gates are used to manipulate the state of a qubit, but in contrast to classical gates, quantum gates always have an equal number of input and output qubits.

The state of a qubit ψ can be represented as a vector $|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$, where α_0 and α_1 are complex numbers (called amplitudes) that satisfy $|\alpha_0|^2 + |\alpha_1|^2 = 1$. Similarly to the classical world, $|0\rangle$ and $|1\rangle$ are the states that are analogue to the classical bits 0 and 1. But unlike in classical computing, a qubit can be in a *superposition* of these two states, where it is both 0 and 1. For example, the state $\alpha_0|0\rangle + \alpha_1|1\rangle$ has a probability of $|\alpha_0|^2$ of being measured in the $|0\rangle$ state and a probability of $|\alpha_1|^2$ of being measured in the $|1\rangle$ state. These states represent quantum wavefunctions and it is the interference of these wavefunctions driving quantum processing. Algorithms are designed such that the amplitude related to the correct answer is amplified while the wrong answers are suppressed.

Another similarity between quantum computing and classical computing is that computations can be expressed as circuits. *Quantum circuits* are a description of a quantum algorithm (e.g. see Fig. 3). Quantum circuits have the input on the left and the output on the right. The horizontal *wires* represent the qubits, and the computation is performed by applying a time ordered set of gates, left to right.

A quantum gate manipulating m qubits is described by a $2^m \times 2^m$ unitary matrix acting on an column vector of length 2^m with entries $\{\alpha_i\}$ where $\sum_{i=0}^{2^m-1} |\alpha_i|^2 = 1$ for $i \in [0, \dots, 2^m - 1]$ representing the m -qubit state $|\phi\rangle = \sum_{i=0}^{2^m-1} \alpha_i |i\rangle$. For example, the *controlled-not* gate (CNOT) acting on two qubits is defined by the following 4×4 matrix:

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \quad (1)$$

Performing the CNOT operation between two qubits c and t in some arbitrary quantum state¹ results in the following output:

$$\begin{aligned} |ct\rangle_{in} &= \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle \\ |ct\rangle_{out} &= \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|11\rangle + \alpha_{11}|10\rangle \end{aligned} \quad (2)$$

Although in this section only the single-target CNOT is discussed, the provided details can be easily generalised for the multi-target CNOT. In Fig. 3 the CNOT gate is depicted as the vertical line connecting two qubits. The filled black dot represents the control qubit, while the \oplus is the target qubit.

¹The state of the qubits is written in a shorthand notation: $|c\rangle|t\rangle = |ct\rangle$.

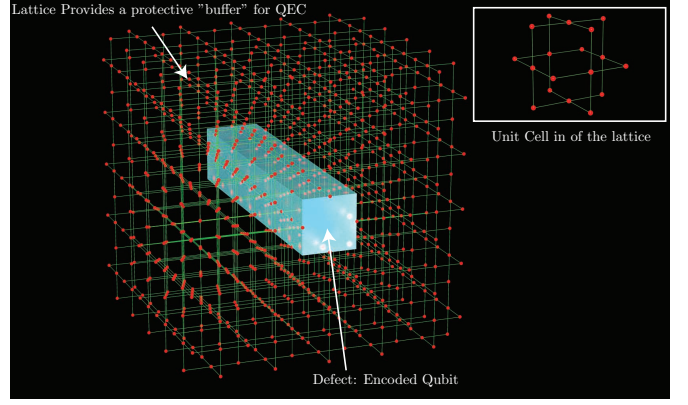


Fig. 1. *Logical volume of the topological cluster.* Each red dot represents a physical qubit, green lines represent links (entanglement between qubits). An encoded piece of information is a rectangular hole that is created by physically removing qubits internal to its boundary, the lattice surrounding the defect provides the error correction "buffer". The insert illustrates a unit cell of the lattice, consisting of 18 qubits.

Reading the value of a bit has a quantum counterpart; measurement. Unlike classical readout, quantum measurement allows us to read out qubits in multiple ways. The standard measurement in quantum computation, referred to as a *Z*-basis measurement, measures if the qubit is in the $|0\rangle$ or $|1\rangle$ state, collapsing (where the wavefunction describing the quantum state discontinuously changes) any superposition inconsistent with the measurement result. Another type of measurement is an *X*-basis measurement, which measures if the qubit is in the $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ state or the $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ state. This type of measurement is valid as these two states are orthogonal (the wavefunctions describing these states have zero overlap). In Fig. 3, the encircled *X* and *Z* are representing these two types of measurements.

B. Topological Quantum Computing

Several quantum architectures are based upon the model of TQC [4]–[6], as this method for computation has QEC integrated by construction. TQC is the preferred method for three primary reasons. 1) It has one of the highest fault-tolerant thresholds of any method of QEC². 2) It is a *local* model of computation, i.e. individual physical qubits in the computer only have to interact with their neighbours. 3) The quantum hardware is only used to prepare a large three-dimensional lattice of connected qubits (the topological cluster), algorithmic implementation is a function of how the lattice is consumed rather than how it is created. Therefore the TQC model is a software driven method of computation.

The specifics of how TQC works is complicated, however the general principal can be explained. The quantum hardware prepares a massive 3D lattice of qubits that are all connected (entangled) to form a single enormous quantum state. The unit cell of this lattice is shown as the insert in Fig. 1. Logical information is introduced and error protected by deliberately

²The Fault-tolerant threshold is the fundamental error rate associated with the physical hardware that must be reached before QEC becomes effective.

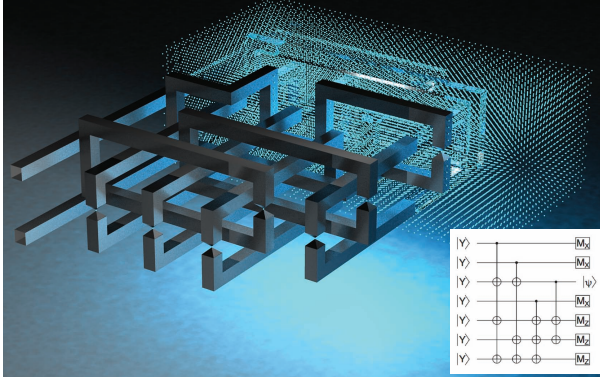


Fig. 2. A topologically protected quantum circuit using defect braiding. The insert illustrates the quantum circuit realised. CNOT gates are achieved via the geometric movement of defect structures to form closed braids. The geometric movement of defects is realised by the selective removal of physical qubits internal to the defect boundaries. The physical resources required are determined from the total volume of the lattice.

creating holes in this lattice. Shown in Fig. 1 is an example of a logically encoded qubit. The information is stored within a hole (defect) in the lattice, this defect is created by simply removing the physical qubits internal to its boundary. In Fig. 1, the defect is the rectangular structure and all physical qubits internal to this structure are simply removed from the lattice. The defect is surrounded by the bulk of the lattice, it is this bulk that provides the error protection of the model. If a defect has a large cross-section and is surrounded by a large "buffer" of the lattice, the information is heavily protected. The details of how this works can be found in [11], [12].

The purpose of computation with this model is to, in a controlled manner, manipulate the shape and movement of the defects within a large lattice produced by the hardware. Illustrated in Fig. 2 is a larger quantum circuit (performing a sub-circuit called state distillation which is required to perform valid operations on encoded information). Instead of one defect, we have enough space to introduce multiple defects, representing multiple encoded qubits. These qubits are then interacted by moving defects around each other. These movements are performed by selecting which *physical* qubits need to be removed from the lattice to form these structures.

A large quantum algorithm is therefore defined via these geometric structures and how they are arranged and embedded within the topological cluster. As the total size of the lattice is directly related so the number of devices within our quantum computer, we clearly want to ensure that large quantum algorithms are classically compacted to occupy the smallest possible volume, reducing resources in the hardware.

III. GATES FOR TOPOLOGICAL QUANTUM COMPUTING

Universal quantum computation must be performed by using a discrete and reduced set of gates (whose direct application is compatible with the underlying encoding). In the context of TQC the chosen universal gate set consists of the CNOT gate and a discrete set of single qubit gates [15], [16].

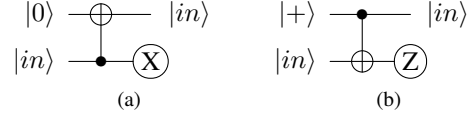


Fig. 3. Quantum teleportation circuits: The arbitrary $|in\rangle$ state is teleported from the lower qubit to the upper qubit.

Single-qubit gates are applied via teleportations to a data qubit, by utilising an ancilla (auxiliary) qubit that is prepared in a specific encoded state. This preparation has to be further decomposed into distillation protocols which act to purify multiple noisy copies of these auxiliary states (as they are introduced/injected into the lattice in a non fault-tolerant manner) into a single highly pure copy that can be coupled without introducing additional noise into our encoded data. A more in-depth discussion of how universality is achieved for TQC is given in [12].

State injection and distillation will not be addressed in this work, because they don't dictate the general framework of the synthesis. Fully error corrected computations are ultimately a large array of CNOTs and identities within the cluster. The synthesis of computation will not affect the universality of the TQC model.

A. The Topological CNOT Gate

The CNOT gate is one of the most used gates, however as complete derivation of the junction CNOT in this work is rather complicated, we instead utilise the teleportation circuits introduced in [11] as an assumed starting point. These defect junctions invoke the circuits illustrated in Fig. 4, where each encoded qubit is represented by a pair of defects (holes in the lattice). In one case (Fig. 4b), a CNOT operation is performed between the input qubit and a freshly initialised qubit in the $|0\rangle$ state, with both outputted. In Fig. 4c two encoded qubits are used as input and at the junction point, a CNOT operation is performed, then one of the qubits is measured in the X basis. Before the junction, a ring encircles the two defects and is required to ensure that the logical information is propagated correctly according to the quantum circuit. These circuits are generalisations of the teleportation circuits in Fig. 3 [11] and have a similar geometric structure.

Given these two junctions and their correspondence to the associated quantum circuits, the derivation of the CNOT is achieved by simply attaching one of the output qubits from circuit one to one of the inputs of circuit two and calculating the resulting circuit identity.

$$|c0t\rangle = a_0|000\rangle + a_1|001\rangle + a_2|100\rangle + a_3|101\rangle$$

$$\text{CNOT} \rightarrow a_0|000\rangle + a_1|001\rangle + a_2|111\rangle + a_3|110\rangle$$

$$\text{Measure 2 in } X \rightarrow a_0|00\rangle + a_1|01\rangle \pm a_2|11\rangle \pm a_3|10\rangle \quad (3)$$

where the \pm signs depend on the measurement result of qubit two and can be corrected.

B. The Identity Gate

Another gate that can be directly implemented is the identity gate. This gate is simply reshaping the geometric structure

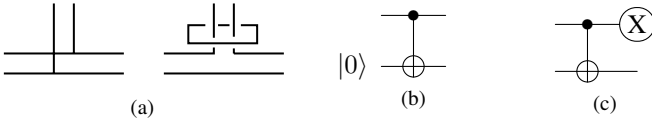


Fig. 4. Teleportation circuits as junctions. Encoded qubits are represented as pairs of defects, CNOT operations, initialisation and measurement occur at the junction point [11]. A defect ring is required in Fig. a) to ensure correct operation. This ring is illustrated more clearly in Fig. 6.

of the defects throughout the lattice. If a pair of defects are utilised to support a single encoded qubit of information then movements must be performed in a manner as to maintain a constant separation of the two defects (this is to ensure that the error correction properties of the code are maintained). Geometrically, defects may be moved in any directions in the 3D lattice provided there is sufficient bulk space³.

IV. SYNTHESIS FOR TQC

TQC synthesis is the process by which a quantum computation (e.g. described using the high-level quantum circuit formalism) is translated into a representation of topological quantum gates. The synthesis process is conceptually similar to classical logical synthesis. The CNOT and the identity can be regarded as the building-blocks necessary for TQC, because as arbitrary single-qubit gates are constructed using CNOT gates, qubit initialisation and measurement in two orthogonal bases (initialisation of encoded qubits are realised by introducing defects through the removal of physical qubits while measurement is realised by terminating a defect by no longer removing physical qubits from the lattice).

Note that the problem formulation is different from the synthesis problem in the field of reversible circuits (Boolean circuits consisting of CNOT gates and other bijective gates). Reversible circuit synthesis [17]–[21] takes either a high-level description or a gate-level net-list of a circuit as the input, and calculates an (optimized) gate-level net-list with an equivalent functionality. In contrast, the presented algorithms generate the arrangement of *defects* of a topological cluster. No optimisations of the gate-level net-list is performed, although such optimisations could further improve the outcome of the algorithms. The major difference between reversible computation and TQC is that, although both make extensive use of the CNOT gate, only in TQC do we require state injection and distillation for achieving the *complete* set of universal gates. Therefore, TQC can implement arbitrary quantum circuits which process quantum states incorporating superposition and entanglement. Reversible circuits process Boolean states, i.e., vectors of 0s and 1s, and are complete within this paradigm.

A quantum computation can be expressed as its constituent set of gates G , and, therefore, for TQC synthesis, G will consist only of CNOT gates. Assuming that all the qubits of a computation are numbered (Q is the set of qubits), a multi-target CNOT $cn \in G$ will be characterised by a qubit

³Bulk space in this context means part of the lattice *not* containing other defects.

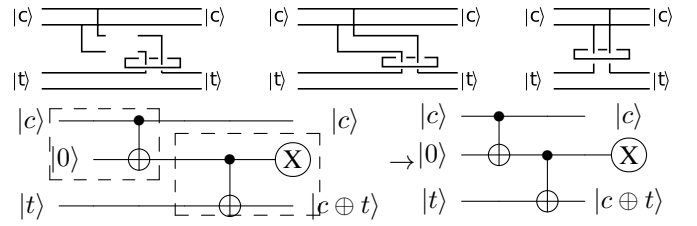


Fig. 5. The topological CNOT and its quantum circuit equivalent after combining the two teleportation circuits from Fig. 4

$control(cn) \in Q$ indicating the control, and a set of qubits $targets(cn)$ indicating the targets.

The structure of the topological QEC is such that encoded qubits form a three-dimensional algorithmic layout. This layout allows nearest neighbour (where encoded qubits only interact with their neighbours) or long range interactions between separate regions of the computer. Utilising long range interactions will have an impact on the ability to optimise the total volume of cluster required for a large computation. Nearest neighbour interactions can be easily implemented by translating quantum algorithms into a two-dimensional field, having as a result a reduced number of possibilities in which a defect can be moved through the cluster. For an operation between two neighbouring encoded qubits to take place, the qubits will be firstly brought near to each other in the field, and will interact afterwards. In this work, in the context of TQC synthesis, the area of the two-dimensional field is the cost function to minimise.

A. Compact Representation of Gates

The TQC synthesis algorithm uses eight primitives (see Fig. 6) to implement the two gates: identity and CNOT. Both the 3D realisation and the two-dimensional representation of the primitives are shown in the figure. Recall that each qubit is represented by a pair of defects running in parallel. The six primitives from Fig. 6a,b perform the identity operation on one qubit (keep one qubit unchanged). The primitive in Fig. 6c keeps two qubits (represented by horizontal and vertical pairs of defects) unchanged. Note that these two pairs of defects appear to touch each other in the two-dimensional representation, but in 3D they pass under each other. From a 3D perspective, the CNOT (Fig. 6d) has a discontinuous qubit as the target which is physically connected to the control. In two dimensions, the control qubit is vertical and the target qubit is horizontal. The ring that encircles the junction within the CNOT is implicit within the 2D representation.

There is a direct mapping from the two-dimensional representation to the topological cluster, and assuming that for a computation all CNOTs and identities are placed on the same *layer*, then the compact two-dimensional representation is adequate for representing arbitrary circuits.

B. The Field of CNOTs

The field of CNOTs is a matrix-like regular structure having the following properties:

- the target qubits of each gate are located on the rows;

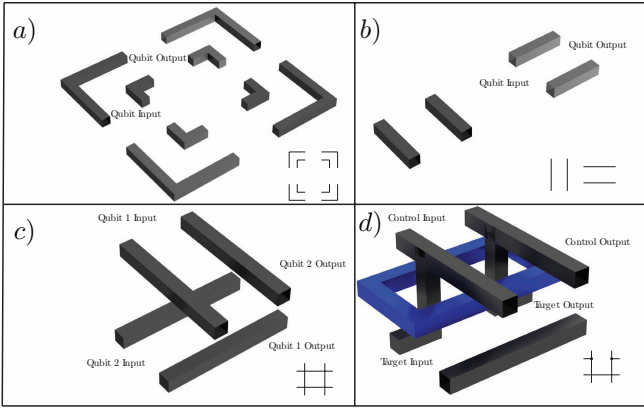


Fig. 6. Compact topological representations of the identity gates and of the CNOT. Each encoded qubit is a pair of defects which always maintain a fixed separation from all other defects (except at junction points), this maintains the error correction strength of the code. The defect ring that encircles the CNOT is implicit in the 2D representation.

- the control qubits of each gate are located on the columns.

This arrangement is possible, because in 3D, the planes supporting the qubits are parallel (see Fig. 6).

The area of the field depends on the width (number of columns) and the height (number of rows), that in turn are influenced by characteristics of the quantum computation, such as the number of gate and qubits. Throughout the following sections a distinction will be made between cluster (a 3D structure with fixed depth) and two-dimensional field. Also, because the cluster volume is linearly related to the field area, when analysing the efficiency of the synthesis, the area will be referred to.

Two alternative two-dimensional fields implementing the 4-qubit, 3-CNOT circuit from Fig. 7a are shown in Fig. 7b and 7c. Note that cn_1 is a multi-target CNOT (it has two targets) and is represented by two single-target CNOTs. The cn_2 operation is implemented on the second column of the field: qubit q_0 is transformed into a control for qubit q_3 . Finally, cn_3 is implemented on the third column. Positions not occupied by CNOTs implement identities. The area is $3 \times 3 = 9$. The field in Fig. 7c implements the same computation in a less efficient way using 12 identity gates, and has an area of $8 \times 6 = 48$.

C. Synthesis Algorithms

The synthesis of quantum computations for a TQC cluster can be automated, and, as previously mentioned, an important aspect is the area of the synthesised field. In this section two synthesis algorithms are presented and evaluated, and both algorithms receive the set of gates G as input, and output a field of CNOTs. The algorithms were designed starting from the assumption, that the output field (topological cluster) will be *consumed* from the left to the right by the TQC hardware. Therefore, the first gate will be the left-most in the field. The computational complexity of the algorithms is linear in the number of gates.

Algorithm 1 is constructing the field on a per-gate basis, and inserts qubits involved in the computation as they are needed.

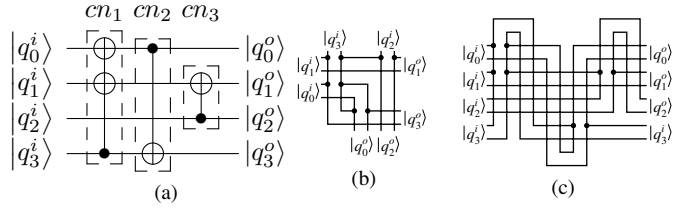


Fig. 7. A circuit and two possibilities to construct the corresponding fields

If a target qubit does not exist in the circuit created so far, it is inserted on a row, and if a control qubit is needed, this will be inserted directly on a column. Also, if for a given gate, that is currently synthesised, an existing target qubit will be used as a control, then the qubit will be moved to the next free column. An existing control qubit needed as a target, will be first moved to the next free row, and then to the next free column.

Algorithm 1 Unbounded: Moving the control downwards

```

1: Initialise an empty field
2: Start with no qubits inserted in the field
3: for all CNOT  $cn \in G$  do
4:   for all  $q \in targets(cn) \cup \{control(cn)\}$  do
5:     if  $q$  does not exist in the field then
6:       insert a row/column for  $q$ 
7:     else
8:       move qubit  $q$  to the next free row
9:     end if
10:  end for
11:  move all  $q \in targets(cn) \cup \{control(cn)\}$  to the next free column
12:  move  $control(cn)$  to the next free row using it as control for  $targets(cn)$ 
13: end for

```

The fields generated by Algorithm 1 typically have the main diagonal occupied, leading to a suboptimal area utilization. With regard to the field area, the worst-case scenario is achieved when the control of each gate was a control, and the number of targets equals the maximum number of qubits: $A_1^w = |G|^2 \cdot (|Q| - 1) \cdot |Q|$.

A second algorithm was designed after noticing the worst-case negative impact of the unbounded construction of the field. If G , the set of gates, is analysed before the synthesis, useful information can be extracted, such as the number of qubits $|Q|$ involved in the quantum computation. The information can be used to initialise the field, thus bounding the number of rows, and also leading to a more efficient and predictable placement of the gates.

The output of Algorithm 2 resembles a *fabric of qubits*, because when $control(cn)$ is transformed into a control and back into a target, the control qubit is moved from a row to a column and back. The transformations are equivalent to the movements of a qubit inside a TQC cluster (see Fig. 6), and all the possible targets are affected because of the control qubit

Algorithm 2 Bounded: Weaving of qubits

- 1: Compute $|Q|$ from G
 - 2: Initialise a field with $|Q| + 2$ rows
 - 3: Place each qubit $q_i \in Q$ on the row $i + 1$
 - 4: **for all** CNOT $cn \in G$ **do**
 - 5: Select the target qubit $control(cn)$
 - 6: Transform $control(cn)$ into a control qubit
 - 7: Continue $control(cn)$ while using it as a control for $targets(cn)$: a) downwards to row $|Q| + 1$; b) to the right one column; c) upwards to row 0; d) to the right one column; e) downwards to row $control(cn) + 1$.
 - 8: Transform $control(cn)$ back into a target qubit
 - 9: **end for**
-

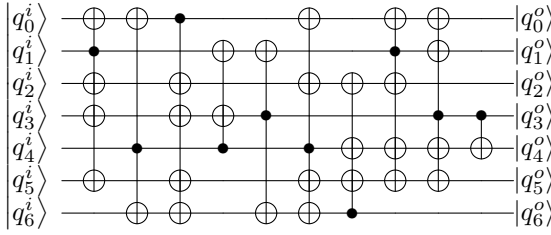


Fig. 8. An sample circuit used for the bounded and the unbounded synthesis

movement pattern.

The algorithm requires two *buffer* rows (the highest and the lowest) to allow the control qubit to change its direction from upwards to downwards, and/or the other way around. Because a complete movement of the control qubit *occupies* three columns of the field, the worst-case area of the output field will be $A_2^w = 3 \cdot (|Q| + 2) \cdot |G|$. However, the area of the output field can be improved by incorporating additional heuristics into the algorithm. For example, information extracted from the set of targeted qubits could be used for the control qubit movement. If, for a given CNOT cn , all the target qubits are placed above the control qubit ($\forall t \in G, control(cn) > t$), then the control will be only moved upwards in the field. The downwards movement is triggered when all the targets are placed below the control ($\forall t \in G, control(cn) < t$).

V. SIMULATION RESULTS

The presented algorithms were implemented, and using Fig. 9 and Fig. 10 the typical output fields generated by each algorithm can be visually compared. The quantum computation for which the fields were synthesised is depicted in Fig. 8: it is applied on 7 qubits, and consists of 10 CNOTs targeting maximum 4 qubits.

For the evaluation of the algorithms, randomly generated circuits consisting entirely of CNOTs were used. The simulation results are presented in Table I. The simulations were parametrised for 10, 100, 1000 and 10000 gates operating on 10, 100 and 1000 qubits. The *MT* column contains the value of the maximum size of $targets(cn)$ per randomly generated CNOT, and the *Columns* and *Rows* columns contain the average number of rows and columns obtained by the algorithms after executing them 100 times for each combination

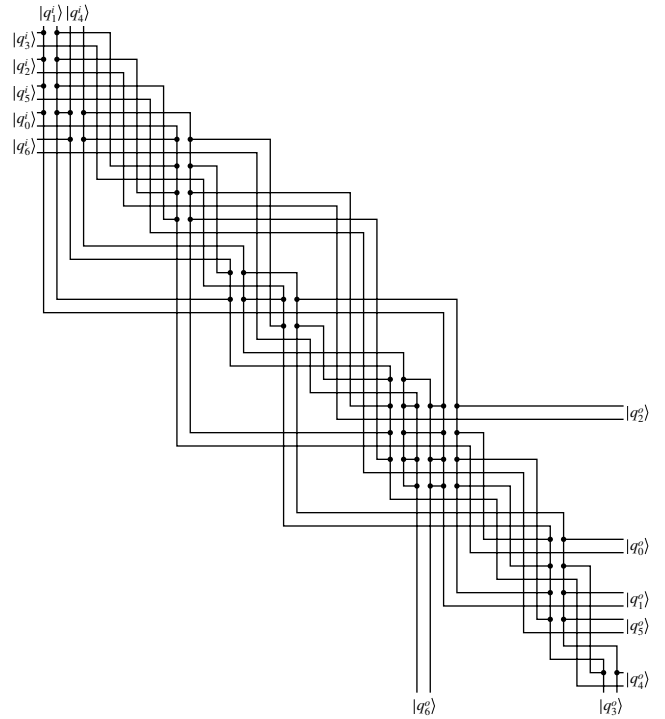


Fig. 9. Output field of the unbounded synthesis

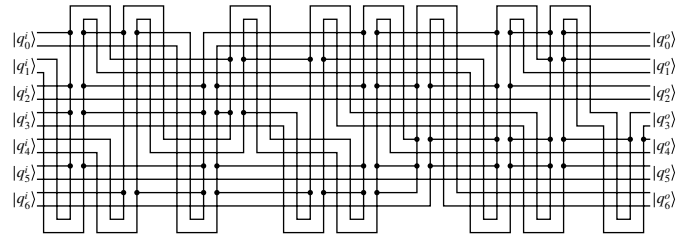


Fig. 10. Output field of the bounded synthesis

of simulation parameters. Algorithm 2 always generates a field with $|G| + 2$ rows, therefore column *Rows* has been omitted from the table. It can be observed that the *MT* parameter is directly influencing the average number of columns and rows obtained through Algorithm 1, but the results of Algorithm 2 are not as strongly affected.

The *Red* column contains the reduction obtained when utilising the bounded synthesis instead of the unbounded algorithms. The unbounded algorithm can also perform better than the bounded algorithm (an example is shown in Figs. 7b and 7c). The simulation results contain such a situation: circuits with 100 qubits and 10 CNOTs having a maximum of 20 targets (see the highlighted entry in Table I). In general, unbounded synthesis could perform better for circuits with less gates and less targets per CNOT.

Overall, the average area of the fields synthesised by the unbounded algorithm is larger compared to the average field area obtained with the bounded synthesis. This effect is particularly significant for large circuits like the ones consisting of 10000 CNOTs.

TABLE I
UNBOUNDED AND BOUNDED SYNTHESIS RESULTS

Q	G	MT	Algorithm 1			Algorithm 2		Red
			Rows	Cols	Area	Cols	Area	
10	10	2	18	13	2.7e+02	17	1.9e+02	1.42e+00
10	10	5	33	19	6.2e+02	18	2.1e+02	2.95e+00
10	10	9	57	30	1.7e+03	19	2.2e+02	7.73e+00
10	100	2	222	160	3.6e+04	159	1.9e+03	1.89e+01
10	100	5	381	238	9.0e+04	171	2.1e+03	4.29e+01
10	100	9	589	341	2.0e+05	179	2.2e+03	9.09e+01
10	1000	2	2292	1645	3.8e+06	1587	1.9e+04	2.00e+02
10	1000	5	3832	2413	9.3e+06	1710	2.1e+04	4.43e+02
10	1000	9	5887	3440	2.0e+07	1786	2.1e+04	9.52e+02
10	10000	2	22780	16389	3.7e+08	15832	1.9e+05	1.95e+03
10	10000	5	38512	24254	9.3e+08	17102	2.1e+05	4.43e+03
10	10000	9	59140	34567	2.0e+09	17855	2.1e+05	9.52e+03
100	10	20	88	20	1.8e+03	20	2.0e+03	9.00e-01
100	10	50	234	77	1.8e+04	20	2.1e+03	8.57e+00
100	10	99	473	194	9.2e+04	20	2.1e+03	4.38e+01
100	100	20	1104	561	6.2e+05	188	2.0e+04	3.10e+01
100	100	50	2559	1284	3.3e+06	194	2.0e+04	1.65e+02
100	100	99	5067	2535	1.3e+07	196	2.0e+04	6.50e+02
100	1000	20	11391	6154	7.0e+07	1868	1.9e+05	3.68e+02
100	1000	50	26425	13666	3.6e+08	1930	1.9e+05	1.89e+03
100	1000	99	50928	25916	1.3e+09	1959	2.0e+05	6.50e+03
100	10000	20	114003	61960	7.1e+09	18676	1.9e+06	3.74e+03
100	10000	50	264204	137056	3.6e+10	19294	2.0e+06	1.80e+04
100	10000	99	509695	259799	1.3e+11	19577	2.0e+06	6.50e+04
1000	10	200	806	88	7.1e+04	21	2.1e+04	3.38e+00
1000	10	500	2183	629	1.4e+06	21	2.1e+04	6.67e+01
1000	10	999	4541	1778	8.1e+06	21	2.1e+04	3.86e+02
1000	100	200	9781	4450	4.4e+07	198	2.0e+05	2.20e+02
1000	100	500	24787	11947	3.0e+08	200	2.0e+05	1.50e+03
1000	100	999	50039	24572	1.2e+09	200	2.0e+05	6.00e+03
1000	1000	200	100963	50491	5.1e+09	1976	2.0e+06	2.55e+03
1000	1000	500	249909	124959	3.1e+10	1990	2.0e+06	1.55e+04
1000	1000	999	499209	249607	1.3e+11	1994	2.0e+06	6.50e+04
1000	10000	200	1015910	512465	5.2e+11	19756	2.0e+07	2.60e+04
1000	10000	500	2514665	1261836	3.2e+12	19886	2.0e+07	1.60e+05
1000	10000	999	5001083	2505044	1.3e+13	19936	2.0e+07	6.50e+05

The execution times of the algorithms were negligible, and since the worst-case area consumption can be computed before running the algorithm, it is easier to decide for a given circuit which algorithm to use. An advantage of the bounded algorithm is that input and output qubits are placed on the same row. This is useful when synthesising sub-circuits of a larger circuit, as the subsequent connections of the sub-circuits may be simpler.

VI. CONCLUSION

The integration of large-scale quantum algorithms with the strict requirements of fault-tolerant, error corrected quantum computation is a neglected area of research. While much research has been performed looking at methods for optimising quantum algorithms, these efforts did not take into consideration the requirements and restrictions of error corrected computing such as TQC. The formulation of synthesis for TQC was presented, the primitives were constructed and two algorithms were designed. An important resource of TQC is the volume of the lattice needed for computation, and hence

the algorithms were evaluated with respect to circuit volumes.

Future work will consist on improving the efficiency of the synthesis. For example, irregular (not rectangular) empty cluster space should be efficiently occupied. The properties of the cluster and how topological circuits are constructed offer significant options for optimisation which are still currently under development.

ACKNOWLEDGEMENTS

SJD and KN are partially supported by the Quantum Cybernetics (MEXT) and FIRST projects, Japan.

REFERENCES

- [1] T. D. Ladd, F. Jelezko, R. Laflamme, Y. Nakamura, C. Monroe, and J. O'Brien, "Quantum computers," *Nature (London)*, vol. 464, p. 45, 2010.
- [2] T. Metodi, D. Thaker, A. Cross, F. Chong, and I. Chuang, "A general purpose architecture layout for arbitrary quantum computations," *Proc. SPIE*, vol. 5815, p. 91, 2005.
- [3] A. Fowler, W. Thompson, Z. Yan, A. Stephens, B. Plourde, and F. K. Wilhelm, "Long-range coupling and scalable architecture for superconducting flux qubits," *Phys. Rev. B*, vol. 76, p. 174507, 2007.
- [4] S. Devitt, A. Fowler, A. Stephens, A. Greentree, L. Hollenberg, W. Munro, and K. Nemoto, "Architectural design for a topological cluster state quantum computer," *New J. Phys.*, vol. 11, p. 083032, 2009.
- [5] R. V. Meter, T. Ladd, A. Fowler, and Y. Yamamoto, "Distributed quantum computation architecture using semiconductor nanophotonics," *Int. J. Quant. Inf.*, vol. 8, p. 295, 2010.
- [6] N. C. Jones, R. V. Meter, A. Fowler, P. McMahon, J. Kim, T. Ladd, and Y. Yamamoto, "A layered architecture for quantum computing using quantum dots," *arxiv:1010.5022*, 2010.
- [7] N. Yao, L. Jiang, A. Gorshkov, P. Maurer, G. Giedke, J. Cirac, and M. Lukin, "Scalable architecture for a room temperature solid-state quantum information processor," *arxiv:1012.2864*, 2010.
- [8] S. Devitt, W. Munro, and K. Nemoto, "The beginners guide to quantum error correction," *arXiv:0905.2794*, 2009.
- [9] A. Kitaev, "Fault-tolerant quantum computation by anyons," *Ann. Phys.*, vol. 303, p. 2, 2003.
- [10] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, "Topological quantum memory," *J. Math. Phys.*, vol. 43, p. 4452, 2002.
- [11] R. Raussendorf, J. Harrington, and K. Goyal, "Topological fault-tolerance in cluster state quantum computation," *New J. Phys.*, vol. 9, p. 199, 2007.
- [12] A. Fowler and K. Goyal, "Topological cluster state quantum computing," *Quant. Inf. Comp.*, vol. 9, p. 721, 2009.
- [13] S. Devitt, A. Stephens, W. Munro, and K. Nemoto, "The optical quantum computer: how big and how fast?" *Proc. SPIE*, vol. 8163, 2011.
- [14] M. Nielsen and I. Chuang, *Quantum Computation and Information*, 2nd ed. Cambridge University Press, 2000.
- [15] S. Bravyi and A. Kitaev, "Universal quantum computation with ideal clifford gates and noisy ancillas," *Phys. Rev. A*, vol. 71, p. 022316, 2005.
- [16] C. Dawson and M. Nielsen, "The solovay-kitaev algorithm," *Quant. Inf. Comp.*, vol. 6, no. 1, p. 81, 2006.
- [17] V. Shende, A. Prasad, I. Markov, and J. Hayes, "Synthesis of reversible logic circuits," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 22, no. 6, pp. 710–722, 2003.
- [18] K. Patel, J. Hayes, and I. Markov, "Fault testing for reversible circuits," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 23, no. 8, pp. 1220–1230, 2004.
- [19] M. Saeedi and I. Markov, "Synthesis and optimization of reversible circuits-a survey," *Arxiv preprint arXiv:1110.2574*, 2011.
- [20] P. Gupta, A. Agrawal, and N. Jha, "An algorithm for synthesis of reversible logic circuits," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 25, no. 11, pp. 2317–2330, 2006.
- [21] R. Wille and R. Drechsler, "BDD-based synthesis of reversible logic for large functions," in *Design Automation Conference, 2009. DAC'09. 46th ACM/IEEE*. IEEE, 2009, pp. 270–275.